XML технологии: стратегия обучения

Ефимов А.А.
Руководитель: Рожина И.В.
Уральский государственный педагогический университет

г. Екатеринбург

Статья затрагивает проблему неактуальности текущих инструментов организации всемирной сети. В статье предлагается набор инструментов для современного варианта (Semantic Web посредствам XML технологий) решения этой проблемы и содержится информация о том, чему именно надо учить студентов и в каком порядке это делать для формирования наиболее полного представления о технологиях XML, а так же эффективного и рационального их применения на практике.

Как известно, компьютер "понимает" только язык электромагнитных сигналов, имеющих два уровня. Один уровень обозначается нулем, другой единицей. Поэтому самый простой способ записи текста в компьютере заключается в том, чтобы закодировать каждый символ текста какой-нибудь последовательностью нулей и единиц. Имея в распоряжении один электромагнитный импульс (далее бит), можно закодировать два символа, обозначив один из них единицей, а другой нулем. С помощью двух битов можно закодировать уже четыре символа. Например, можно обозначить букву А комбинацией 00, букву Б — 01, букву В — 10, а букву Г — двумя единицами 11. Три бита дают возможность закодировать восемь символов и т. д.

Обычные английские тексты используют около сотни символов: заглавные и строчные буквы латиницы, цифры, знаки препинания, некоторые математические символы. Для их кодирования применяют комбинации из семи символов, набор которых давно стандартизирован. Этот стандарт называется ASCII (American Standard Code for Information Interchange). Число семь не всегда удобно для компьютера, в машине лучше оперировать степенями двойки. Поэтому в коде ASCII обычно применяют комбинации не из семи, а из восьми нулей и единиц (далее байт), начиная каждую комбинацию с нуля. В стандарте ASCII с использованием восьми разрядов первая буква английского алфавита А кодируется комбинацией 01000001, вторая буква В— комбинацией 01000010, а, например, знак процента % — комбинацией 00100101.

Многие языки, в том числе и русский, требуют записи дополнительных символов — букв кириллицы, символов с диакритическими знаками. Такие символы просто добавляются к набору ASCII. Для их записи используют комбинации из восьми нулей и единиц, начинающиеся с единицы. Разработаны международные стандарты для кодирования дополнительных символов. Символы европейских алфавитов с диакритическими знаками образуют кодировку ISO8859-1, часто называемую Latin 1. Буквы греческого алфавита попали в кодировку ISO8859-2, обычно определяемую как Latin 2.

Для кодирования кириллицы разработана международная кодировка ISO8859-5, но она редко применяется в России. Гораздо чаще используются альтернативные. На русифицированных компьютерах, управляемых операционными системами MS Windows, применяется кодировка CP1251, разработанная корпорацией Microsoft. На машинах, работающих под управлением операционных систем семейства *nix, наряду с CP1251 используется кодировка KOI8-R. Со времен MS DOS осталась кодировка CP866. Машины Apple Macintosh применяют кодировку MacCyrillic.

Это многообразие кириллических кодировок приводит к дополнительной нагрузке на пользователя. Для того чтобы правильно прочитать кириллический текст, он должен знать, в какой кодировке текст был записан, или перебрать несколько кодировок в своем текстовом редакторе.

Итак, для записи простого, как говорят, "плоского" (flat) текста, применяются байтовые кодировки: один символ — один байт. Это удобно для передачи текста. На любом компьютере есть средства для чтения плоского текста. Если знать его кодировку, то текст будет правильно прочитан. Мировое компьютерное сообщество решило освободиться и от этого условия и перейти на двухбайтовую шестнадцатиразрядную кодировку, позволяющую закодировать 65 536

символов. Такая кодировка называется Unicode. В ней перед каждым кодом Latin 1 стоит нулевой байт, перед каждым кодом кириллицы — байт 00000100 и т.д. Для текстов, записанных в Unicode, нет необходимости указывать кодировку, но сам текст занимает ровно в два раза больше места в файле.

Компромиссное решение предоставляет кодировка UTF-8 (Unicode Transfer Format). В ней символы, входящие в ASCII, кодируются одним байтом, национальные символы большинства языков — двумя байтами, зато менее распространенные символы кодируются тремя байтами.

Часто возникает необходимость выделения каких-то фрагментов текста курсивом, подчеркиванием, жирным шрифтом или каким-то другим способом. Иногда надо увеличить размер шрифта, изменить его начертание, вставить в текст буквицу, формулу, какое-то изображение. У байтовых кодировок и у Unicode нет таких возможностей, ведь для курсива или подчеркнутых символов нужно задавать дополнительные коды. Для выделения фрагментов текста надо вводить новые средства. Развитие текстовых редакторов пошло двумя путями.

Первый путь — применение графических текстовых редакторов, таких как MS Word. Они используют не кодировку символа, а описание его графического изображения. Полученное описание записывается в файл специального формата, прочитать который может только такой же текстовый редактор.

Второй путь — внести в текст пометки, указывающие на особенности того или иного фрагмента текста. Этот способ давно применяется в полиграфии. При подготовке документа к печати его рукописный вариант испещряется корректурными знаками и цветными пометками, указывающими наборщику, какой шрифт выбрать при наборе текста. Именно по этому пути пошла Web-технология, успешно воплотив его в гипертекстовом языке разметок HTML (Hypertext Markup Language). В этом языке пометки, называемые тегами (tags), записываются в угловых скобках, чтобы отличить их от символов самого текста. Некоторые теги языка HTML показаны в листинге.

Листинг. Текст с тегами языка HTML

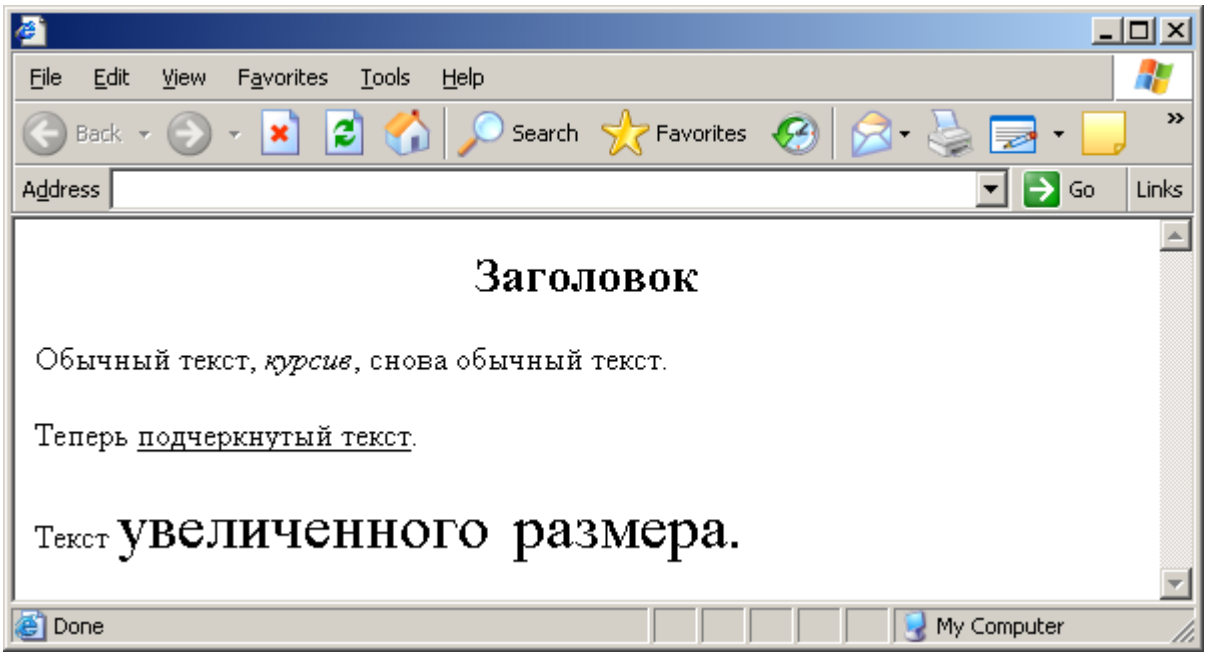
```
<html>
<body>
<h2 align=center>Заголовок</h2>
<p>Обычный текст, <i>курсив</i>, снова обычный текст.</p>
<p>Теперь <u>подчеркнутый текст</u>.</p>
<p>Текст <font size=+3>увеличенного размера.</font></p>
</body>
</html>
```

В листинге тег <h2> (header) показывает, что дальше идет заголовок второго уровня. Параметр align=center тега <h2> предписывает размещение заголовка посередине окна браузера. Тег <i> (italic) означает, что дальше пойдет курсив, а тег <u> (underline) — что дальше надо выводить текст с подчеркиванием. Тег <p> (paragraph) начинает новый абзац. Тег <font> изменяет шрифт текста, в частности параметр size=+3 увеличивает размер шрифта на три единицы.

Как видите, многим тегам языка HTML соответствуют закрывающие теги, отмеченные наклонной чертой. Они отменяют действие открывающих тегов, возвращая текст к первоначальному виду. Сам текст записывается между тегами <body> и </body>, а весь документ — между тегами <html> и </html>.

Текст с тегами языка HTML "понимают" все браузеры. На рисунке показано, как выглядит текст листинга в окне браузера Internet Explorer. Язык разметок HTML прост и нагляден, но у него есть два существенных ограничения. Во-первых, набор тегов HTML строго фиксирован, его нельзя расширить или изменить. Все браузеры должны таким образом интерпретировать одинаковые теги, чтобы пользователь, написавший текст с разметками HTML, был уверен, что этот текст будет одинаково выглядеть во всех браузерах. Во-вторых, теги языка HTML показывают только визуальную разметку, внешний вид документа, но ничего не говорят о его структуре.

Например, заголовки следует помечать одинаковым тегом, например тегом <h2>, даже если все они одного уровня.



Текст HTML в окне браузера

Простота языка HTML перевесила все его недостатки. Она привела к взрывному росту числа Web-сайтов и авторов многочисленных Web-страничек. Обычные пользователи компьютеров ощутили себя творцами, получили возможность заявить о себе, высказать свои мысли и чувства, найти в Интернете своих единомышленников.

Ограниченные возможности языка HTML быстро перестали удовлетворять поднаторевших разработчиков, почувствовавших себя профессионалами. Введение таблиц стилей CSS (Cascading Style Sheet) и включений на стороне сервера SSI (Server Side Includes) лишь ненадолго уменьшило недовольство разработчиков. Профессионалу всегда не хватает средств разработки, он постоянно испытывает потребность добавить к ним какое-то свое средство, позволяющее воплотить все его фантазии.

Такая возможность есть. Еще в 1986 году стал стандартом язык создания языков разметки SGML (Standard Generalized Markup Language), с помощью которого и был создан язык HTML. Основная особенность языка SGML заключается в том, что он позволяет создать новый язык разметок, определив набор тегов создаваемого языка. Каждый конкретный набор тегов, разработанный по правилам SGML, снабжается описанием DTD (Document Type Definition) — *определением типа документа,* разъясняющим связь тегов между собой и правила их применения. Специальная программа — драйвер принтера или SGML-браузер — руководствуется этим описанием для печати или отображения документа на экране дисплея.

В это же время выявилась еще одна, самая важная область применения языков разметки — поиск и выборка информации. В настоящее время, подавляющее большинство информации хранится в реляционных базах данных. Они удобны для хранения и поиска информации, представляемой в виде таблиц: анкет, ведомостей, списков и т. п., — но неудобны для хранения различных документов, планов, отчетов, статей, книг, которые не могут быть представлены в виде таблиц. Между тем, человечество накопило несметное количество таких документов. Большинство из них уже переведено в электронную форму, как правило, в байтовых кодировках. Очень часто ставится задача выбрать какие-то определенные сведения из таких документов: адреса, фамилии, даты, решения, резюме.

Тегами языка разметки можно задать структурную, а не визуальную разметку документа, разбить документ на главы, параграфы и абзацы или на какие-то другие элементы, выделить важные для поиска участки документа. Например, можно определить тег <address> и помечать им все адреса, встречающиеся в тексте. Легко написать программу, анализирующую размеченный такими тегами документ и извлекающую из него нужную информацию. В нашем примере анализирующая программа просто извлечет весь текст, заключенный между тегами <address> и </address>. Язык SGML оказался слишком сложным, требующим тщательного и объемистого описания элементов создаваемого с его помощью языка. Только одна его спецификация содержит

более пятисот страниц. Он применяется лишь в крупных проектах, например, для создания единой системы документооборота крупной фирмы. Скажем, man-страницы (справочная система) операционной системы Solaris Operational Environment написаны на специально сделанной реализации языка SGML.

Золотой серединой между языками SGML и HTML стал язык XML (extensible Markup Language) — расширяемый язык разметок. Это подмножество языка SGML, избавленное от излишней сложности, но позволяющее разработчику Web-страниц создавать свои собственные теги. Язык XML достаточно широк, чтобы можно было создать все нужные теги, и достаточно прост, чтобы можно было быстро их описать.

Важно понимать, что XML – действительное подмножество SGML, это означает, что любой XML документ будет одновременно, и SGML документом и будет распознаваться SGML браузером.

Разработка XML началась в 1996 году. Ею занимается общественная организация W3C - World Wide Web Consortium [http://www.w3c.org/]. Сведения об XML собраны на странице [http://www.w3c.org/XML/]. В 1998 году консорциум выпустил спецификацию XML версии 1.0. Она постоянно совершенствуется, последний вариант спецификации всегда находится по адресу [http://www.w3c.org/TR/rec-xml]. Появление новой версии Unicode 3.0 вызвало необходимость перевода на нее XML. Консорциум W3C начал разработку второй версии XML 1.1. В данный момент она находится в завершающей стадии разработки. Её текущее состояние можно посмотреть по адресу [http://www.w3c.org/TR/xml11]. Уже создана версия Unicode 4.0, поэтому, видимо, выпуск XML 1.1 опять будет отложен.

На данный момент массив XML технологий является одним из самых перспективных способов организации данных в сети, да и самой всемирной паутины [http://www.w3.org/2001/sw]. Тому есть одна достаточная причина – язык XML полностью формализован, то есть, он трактуется однозначно, согласно спецификациям и кто бы не обрабатывал конкретный документ – результат будет всегда одинаков, но в то же время, наряду с полной формализацией данный язык элементарно обрабатывается и анализируется как компьютером, так и человеком.

Правила написания документа XML не сложны. Они подробно рассматриваются в *первом модуле*. Но с каждой разновидностью документов XML надо связать ее описание DTD (Document Type Definition) или какое-то другое описание структуры и способа разметки документов. Такое описание часто бывает сложнее самого документа. Оно выполняется на специальном языке описаний, значит, надо изучить и этот язык. Разбору этих языков будут посвящены *второй* и *третий модули*.

Одного описания документа недостаточно для успешной работы с ним. Нужны еще средства поиска информации в документе, перекрестные ссылки одних частей документа на другие. Этому посвящены следующие *модули*.

Но и этого мало для работы с документами. Необходимо уметь быстро извлекать из документа нужную информацию, уметь преобразовывать документ и представлять его в различных видах. *Восьмой* и *девятый модули* подробно объясняют способы решения этих задач, предложенные технологией XML.

Наконец, в *девятом* и *десятом модулях* раскрывается практическая суть технологии XML, показываются методы работы с документами XML, объясняется, как создаются программы-обработчики этих документов.

Стратегия обучения

В *первой группе* модулей рассматриваются конструкции языка XML и его реализаций, непосредственно связанных с технологией XML.

Первый модуль описывает конструкцию самого документа XML. Из него становится ясно, из чего состоит документ XML, как правильно записать элементы XML и научитесь корректно составлять хорошо оформленные документы XML.

К каждому хорошо оформленному документу XML должно быть приложено его описание. Его можно сделать по-разному.

Из *второго модуля* станет ясно, как делать описание документа на языке DTD (Document Type Definition). Этот язык, пришедший в XML из стандарта SGML, оказался не совсем подходящим для описания документов XML. Он недостаточно подробен, у него не хватает средств полного

описания всех конструкций XML и сейчас он постепенно заменяется другими языками. Тем не менее, его надо знать, потому что уже тысячи документов снабжены такими описаниями.

В *третьем модуле* описан наиболее мощный язык описания документов XML — язык XSD (XML Schema Definition Language). Этот язык сам является реализацией XML. Описание документа, сделанное на нем, само будет документом XML. Эта особенность языка XSD позволяет автоматически обрабатывать описания как обычный документ XML.

Успех языка HTML во многом определен возможностью легко создавать гиперссылки на другие документы или на иные точки того же самого документа. Такая возможность есть и в технологии XML. Ее предоставляет язык XLink (XML Linking Language). Возможности языка XLink гораздо шире, чем возможности тега <a> языка HTML. Им посвящен *четвёртый модуль*.

Иногда надо сделать ссылку не на определенный элемент документа, а на что-нибудь вроде "третьего абзаца пятого параграфа четвертого раздела договора". Это легко сделать на языке XPointer, также входящем в технологию XML. Изучению этого языка и способам его практического применения посвящен *пятый модул*ь.

Шестой модуль также познакомит со ссылками, но не на одно конкретное место документа, а, например, на некоторое множество элементов или на определенное слово, где бы оно ни находилось. Такие ссылки делаются на специальном языке адресации XPath. Это очень мощный язык, используемый во многих других языках, входящих в технологию XML. Область его применения все время расширяется и уже сейчас его просто необходимо знать для понимания особенностей большого числа конструкций XML.

Всем известно, какую большую роль играет язык SQL в работе с базами данных. Поскольку сейчас XML все чаще применяется для хранения документов в базах данных, нужны какие-то средства для быстрого и точного извлечения данных самого разного вида из документов XML. Такие средства предоставляет новый язык XQuery, только что прошедший стадию предварительной разработки. Его подробному изложению посвящен *седьмой модуль*.

На этом *первая группа* модулей, посвященная описанию базовых средств технологии XML, заканчивается. Усвоение материала модулей первой группы, даёт уверенность в понимании сути технологии XML.

Вторая группа модулей описывает средства обработки готовых документов XML.

В *восьмом модуле* подробно рассмотрены конструкции языка преобразования документов XML, называемого XSLT (eXtensible Stylesheet Language for Transformations). Этот язык позволяет на базе одного документа XML автоматически построить другой, может быть, имеющий совершенно иную структуру, или выделить какие-то части документа и составить на их основе новый документ или даже несколько документов. Если есть необходимость собирать и размножать сведения, содержащиеся в документах XML, то язык XSLT – решение проблемы.

Каждый документ XML, в конечном счете, должен быть распечатан или выведен на экран дисплея, сотового телефона, пейджера или даже на какое-то голосовое устройство. Для этого он должен быть предварительно отформатирован в расчете на конкретное представление в устройстве вывода. Такое представление можно сделать на языке XSL-FO (eXtensible Stylesheet Language Formatting Objects), описанию которого посвящен *девятый модуль*.

Заключительные *модули — девятый* и *десятый*, предназначены для любителей программирования, которые хотят понять алгоритмы обработки документов XML. В них изложены методы разбора, анализа и обработки документов XML.

В *десятом модуле* рассматривается обработка документов, основанная на событиях, а в *одиннадцатом* — обработка, основанная на построении дерева документа в оперативной памяти. Освоив материал этих модулей, можно написать собственные программы-обработчики документов XML.

Технологии XML развиваются очень быстро и бурно. В настоящее время они еще сравнительно просты и обозримы в пределах одной книги. Через несколько лет XML обрастет сложными и громоздкими конструкциями, овладеть которыми будет нелегко. Чем раньше приступить к изучению конструкций XML, тем легче в будущем будет следить за развитием данной технологии, и быть в курсе всех событий, связанных с ней. Данный факт даёт право утверждать, что технологии XML должны изучаться обязательно наряду с обучением программированию и Web-дизайну.